Since Go's release in 2009 many companies (besides Google, of course) have used the language to build cool stuff. Join Gustavo Niemeyer from Canonical, Keith Rarick from Heroku, Evan Shaw from Iron.io, and Patrick Crosby from StatHat as they share their first-hand experience using Go in production environments

About Andrew Gerrand

View full profile

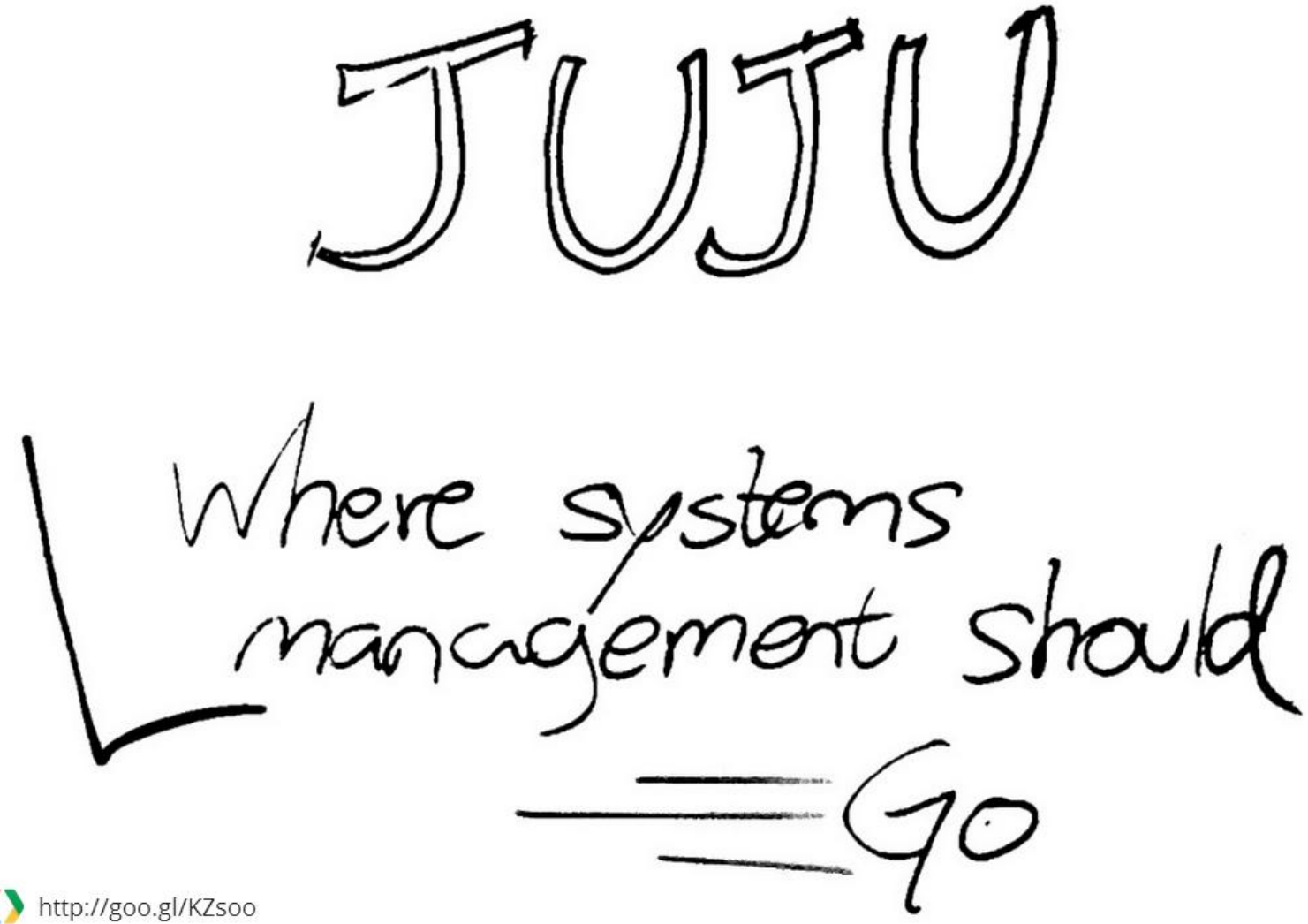
Andrew is a core developer of the Go Programming Language based at Google Sydney. His mission is to teach the world to love Go as much as he does.

# Go in production

Andrew Gerrand - Google
Gustavo Niemeyer - Canonical
Keith Rarick - Heroku
Evan Shaw - Iron.io
Patrick Crosby - StatHat


# Gustavo Niemeyer

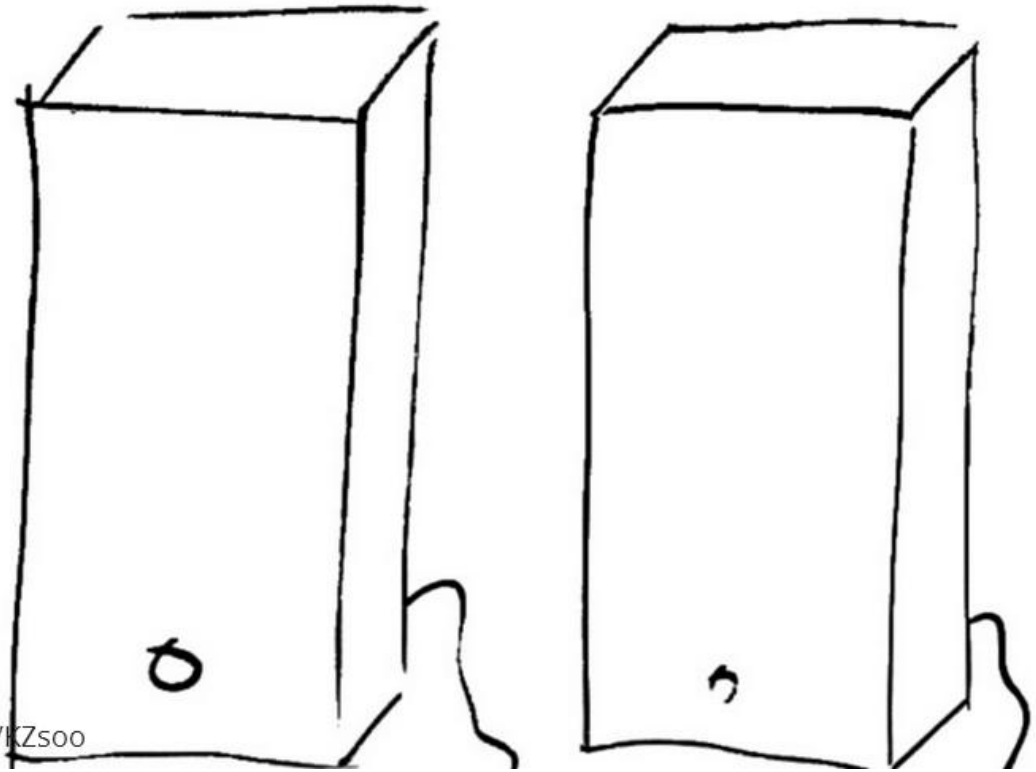
Canonical


JUJU
Where systems management should
Go

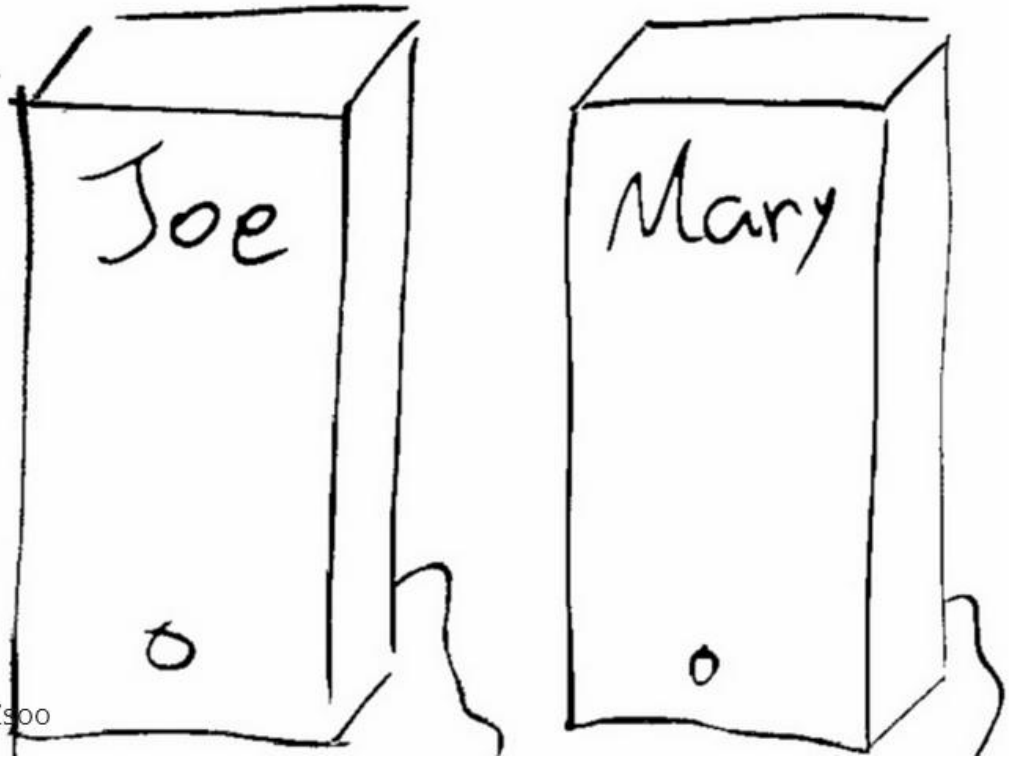
The old style of management.

Was a bit like this...

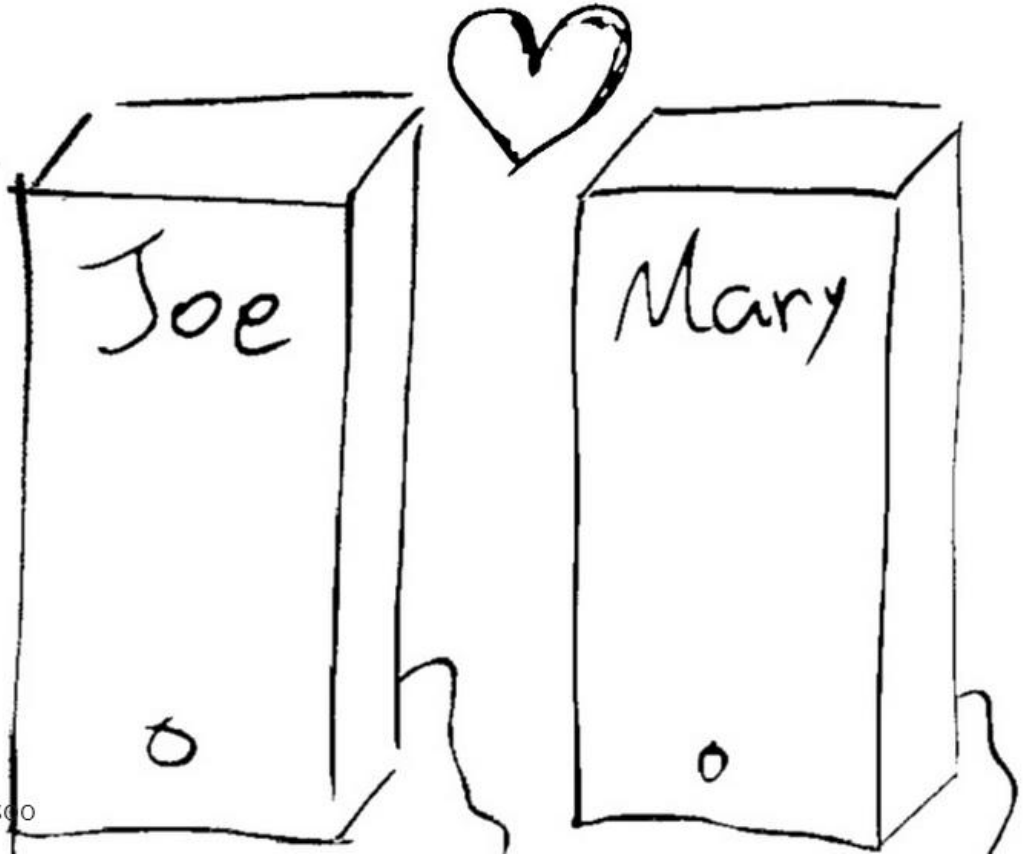
# People had machines...



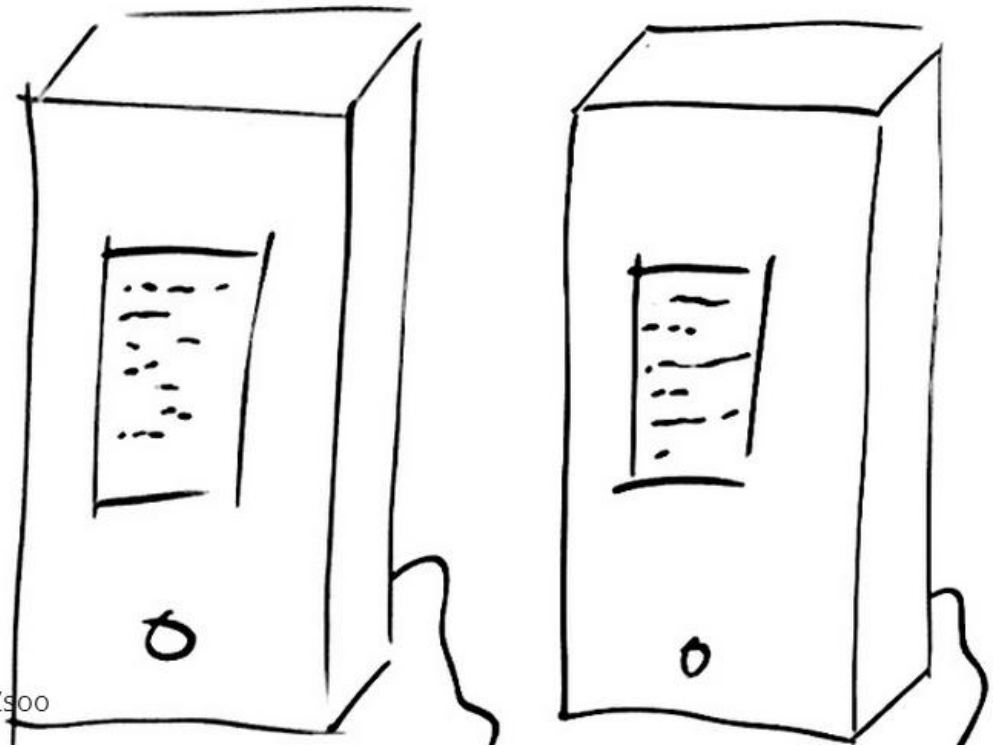
# They gave them names...



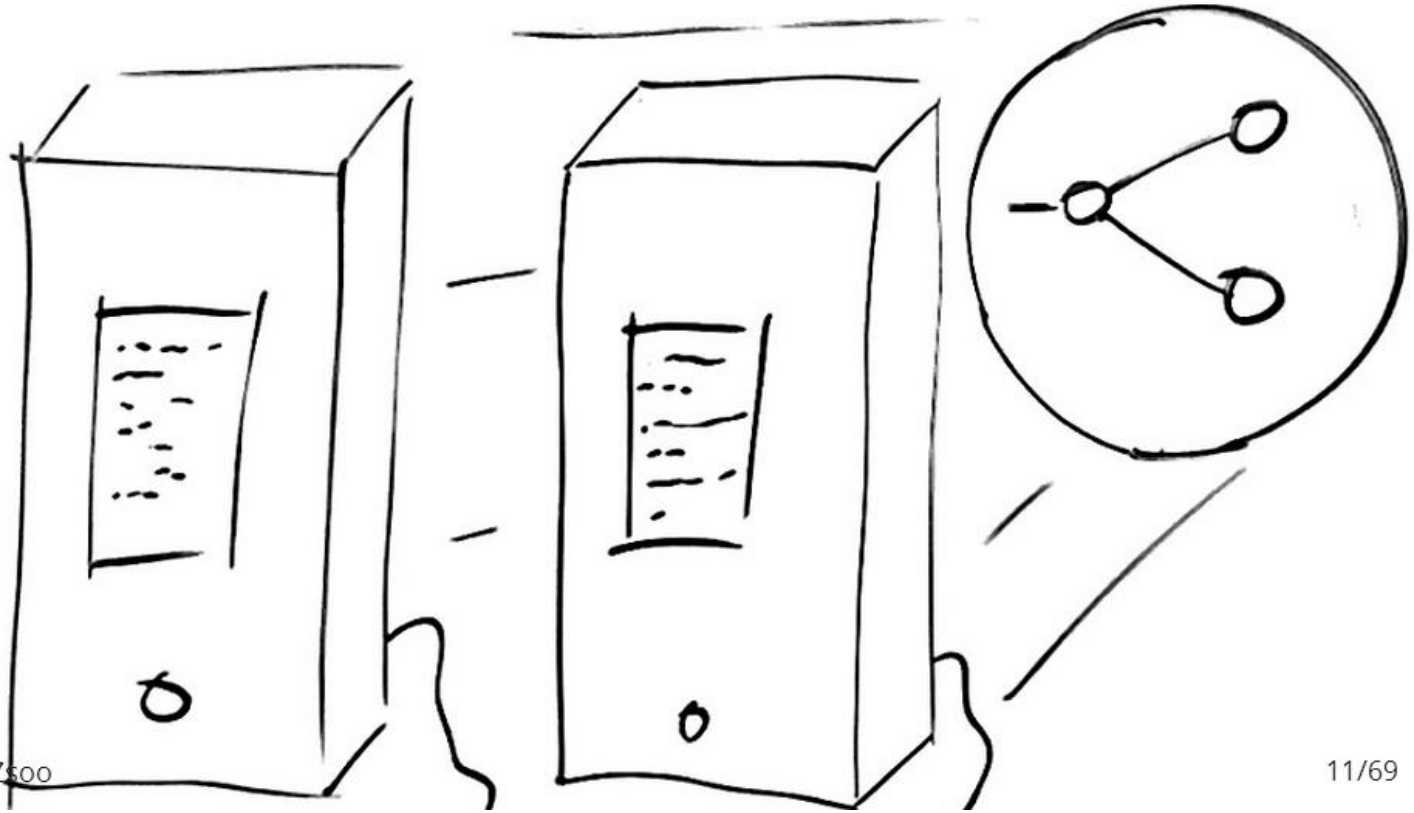
# ... and loved them.



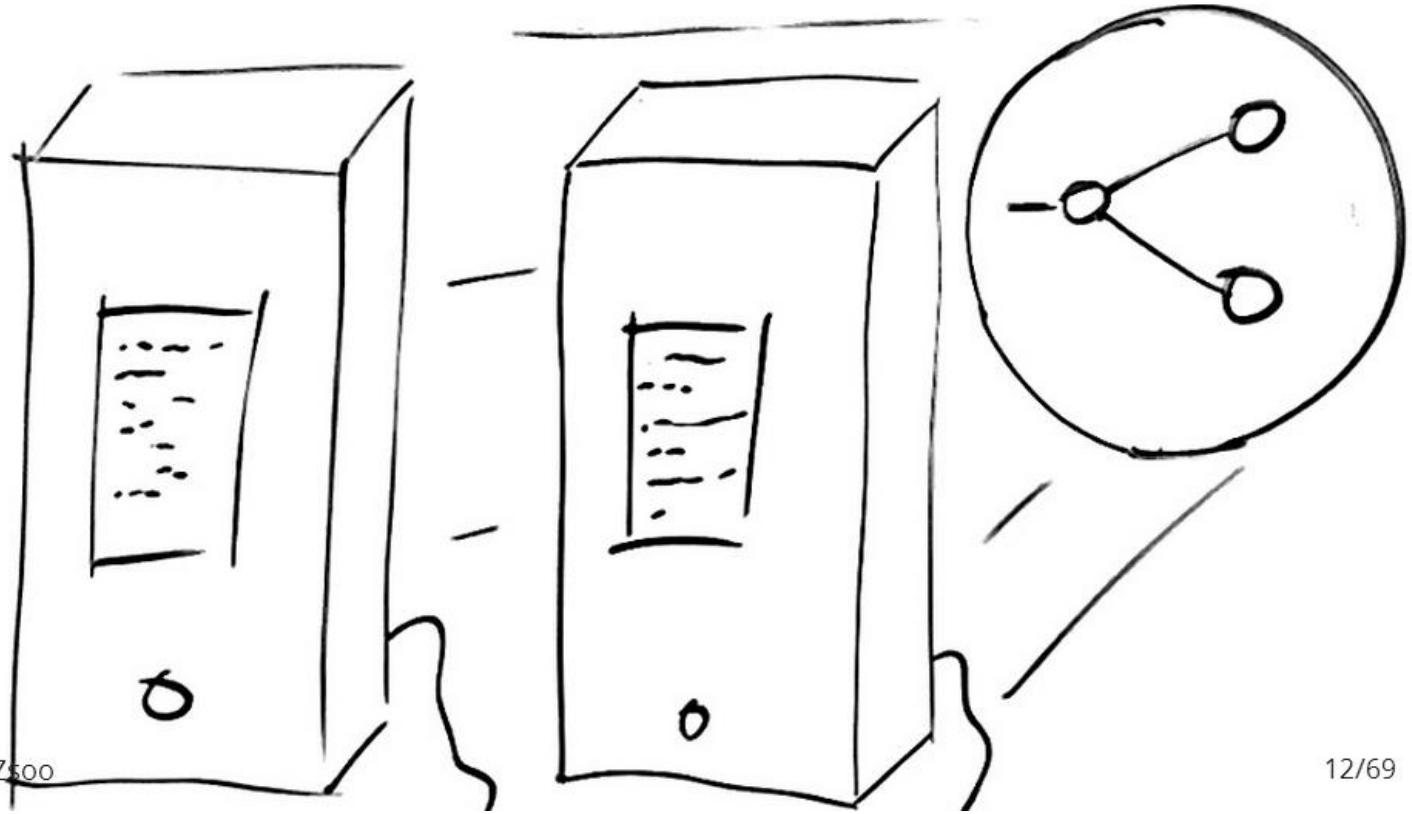
# These machines had config files...

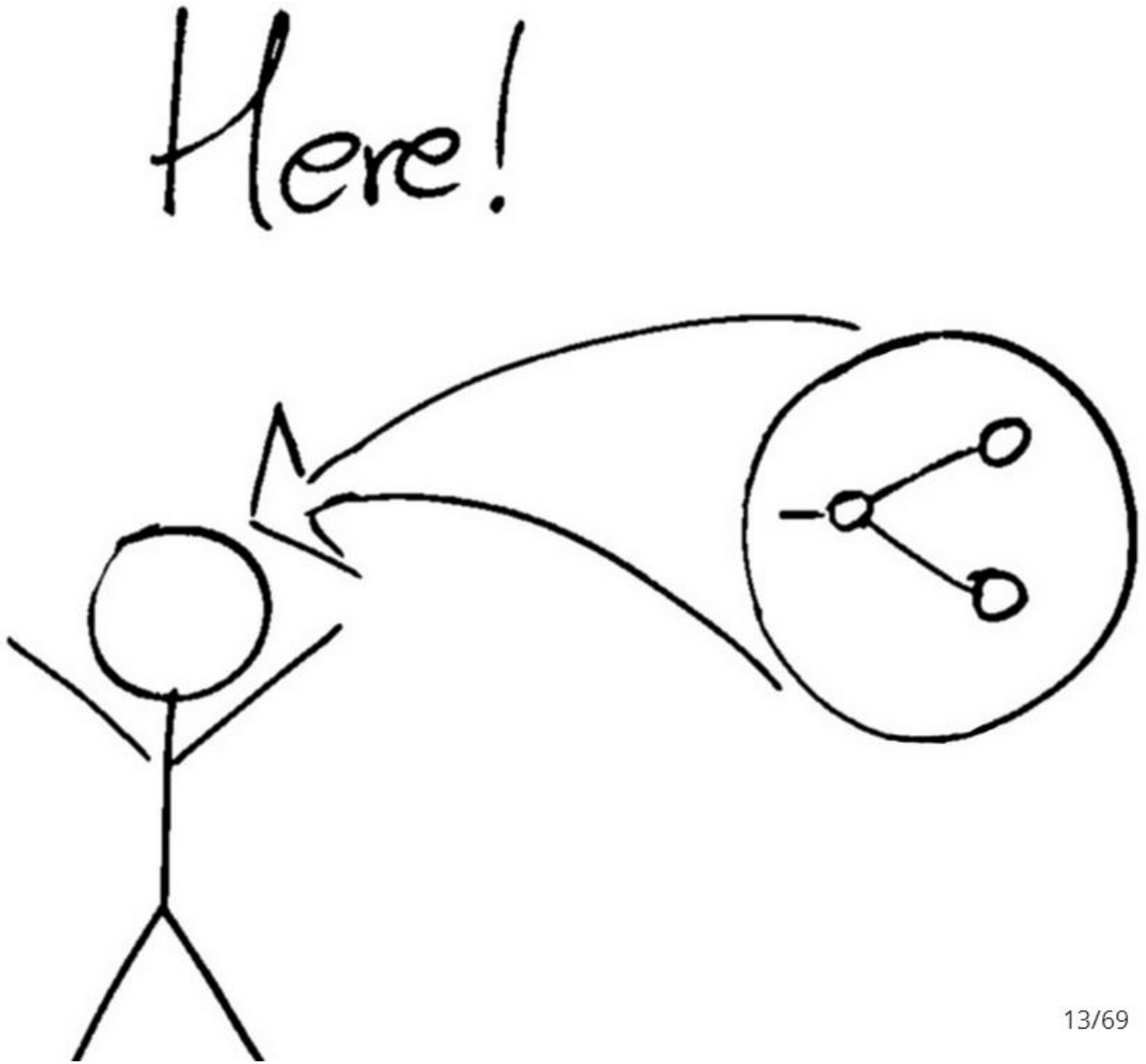


...to assemble a service topology.



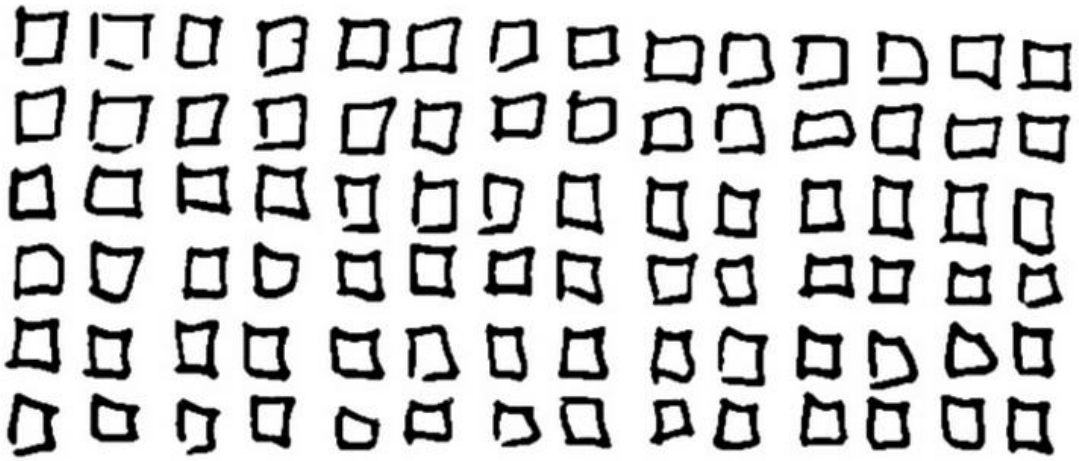
# That topology was stored...



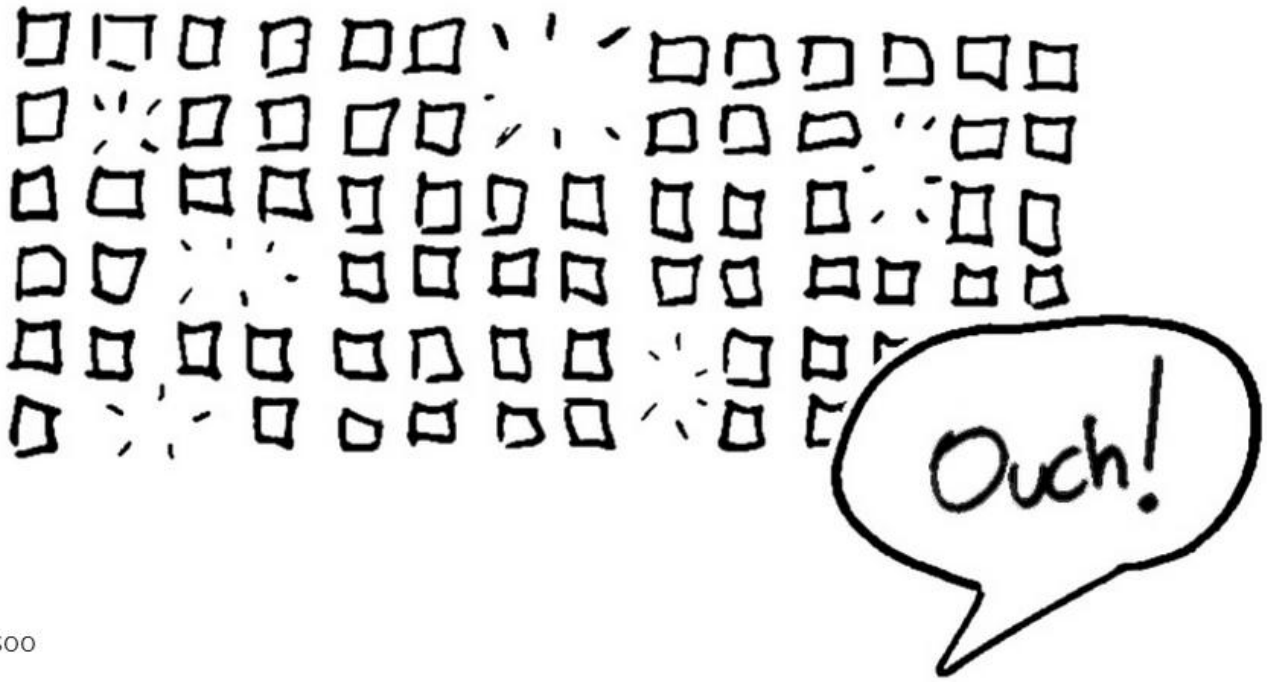

Here!

Then, there were a few issues...

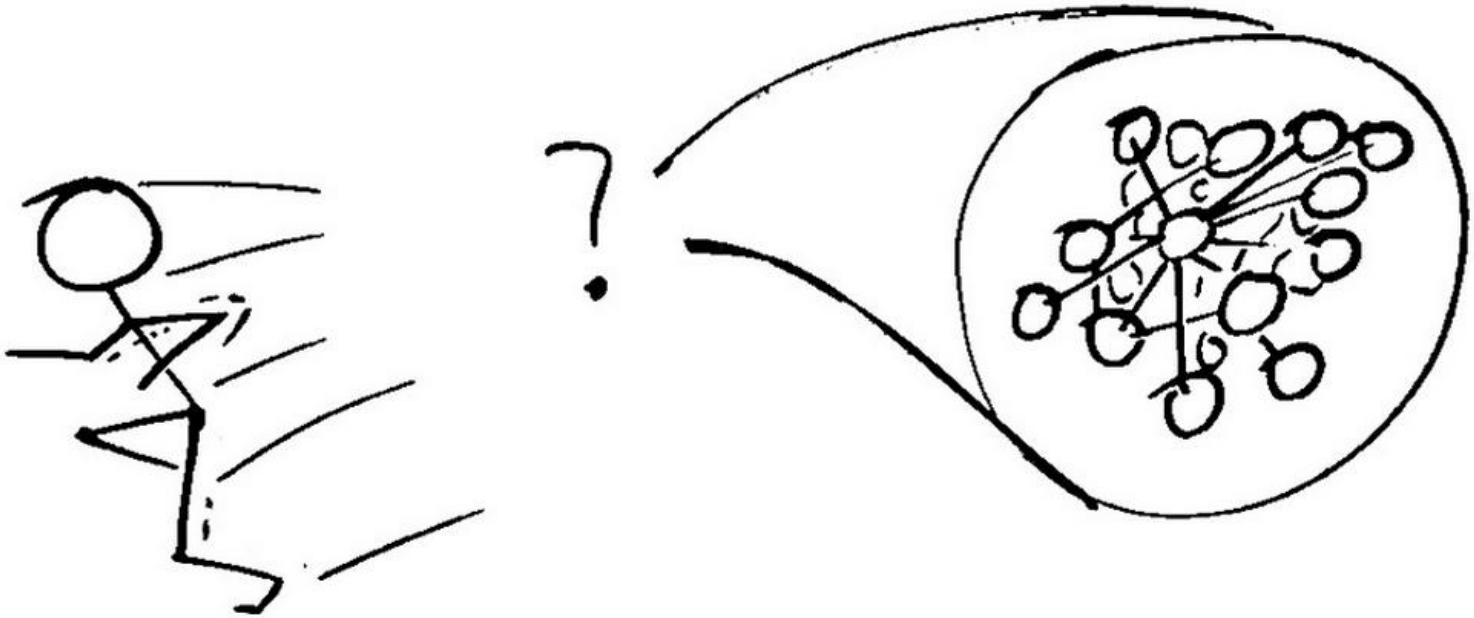
It become a bit hard to give names...



# ...and tragic.

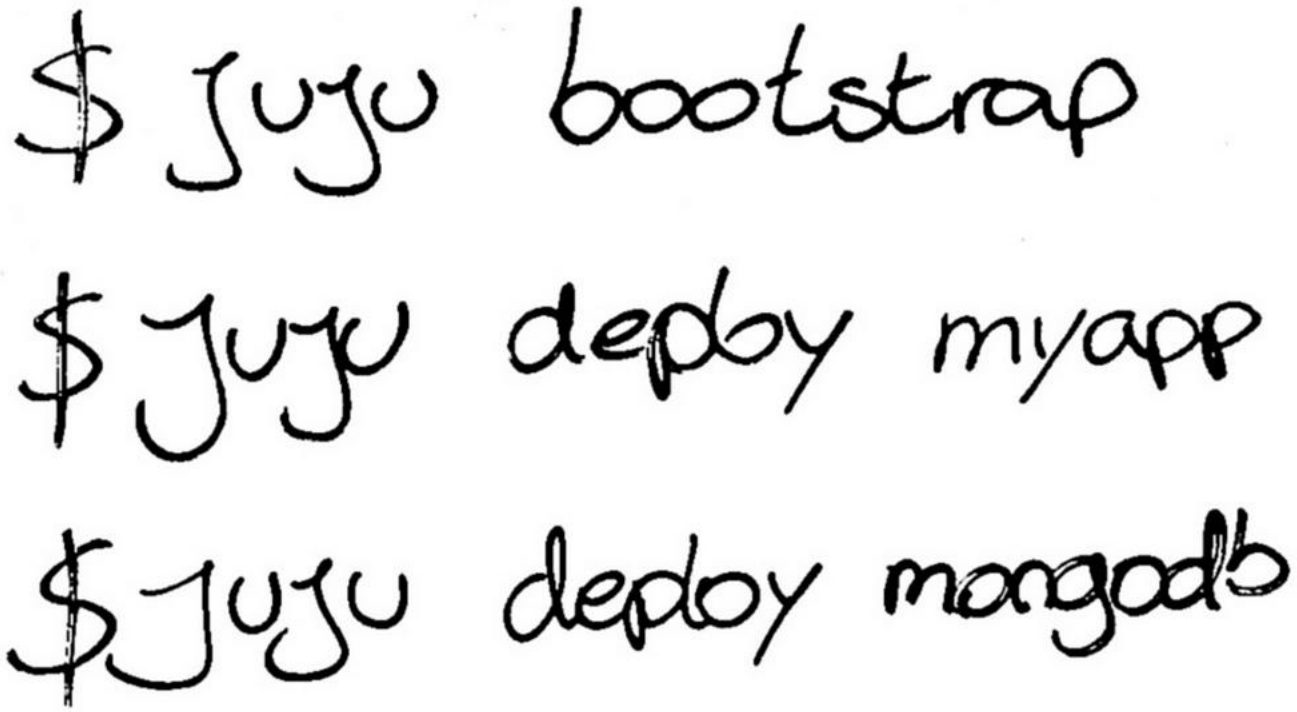


# The topology...



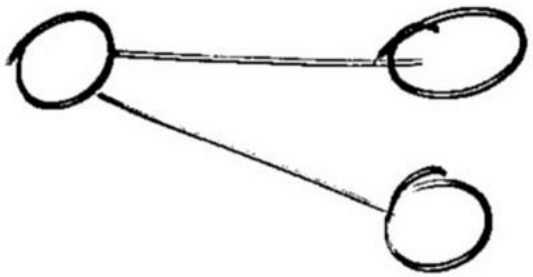
How to...

- Collaborate
- Reuse
- Manage.

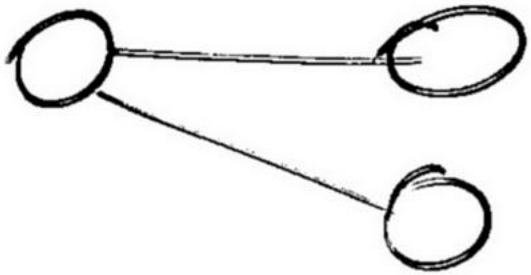

$ juju bootstrap
$ juju deploy myapp
$ juju deploy mongodb

$juju add-relation app mongo

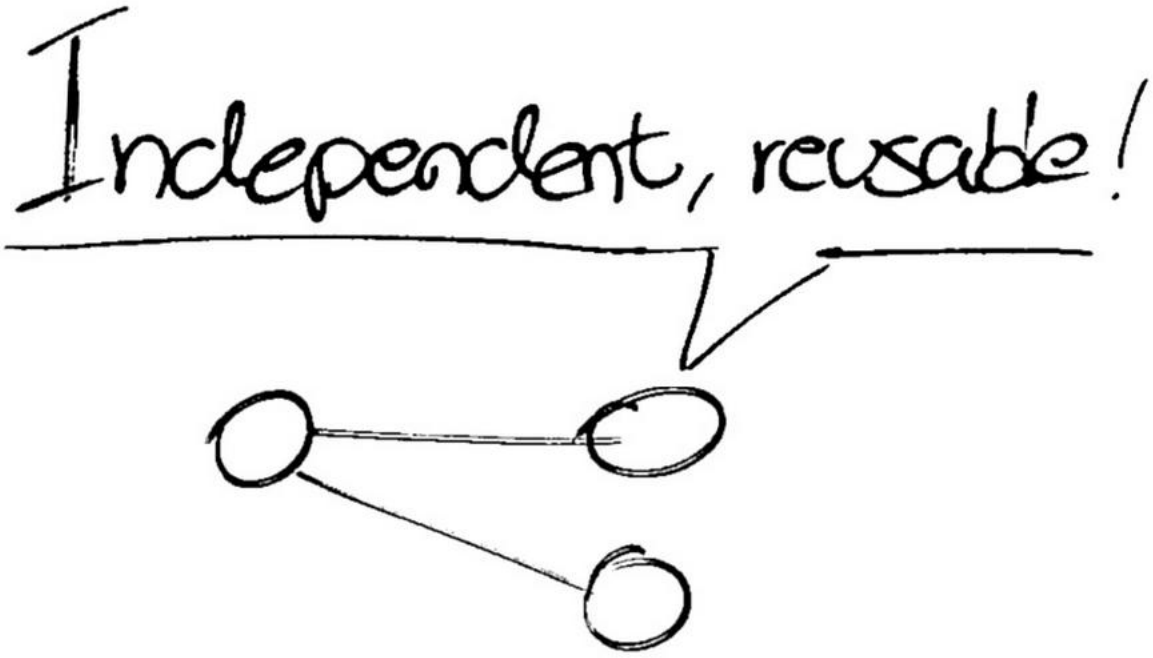
$juju add-unit mongo

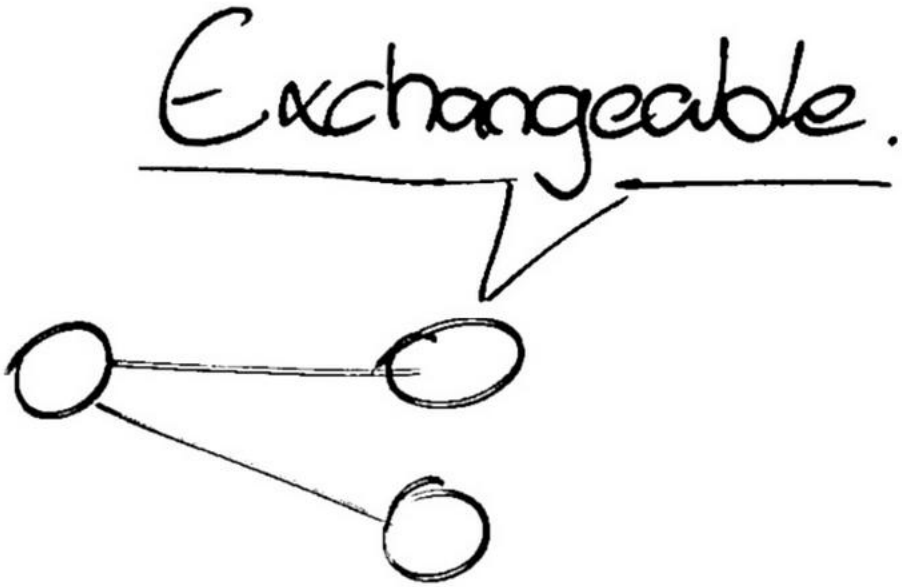


$juju add-unit mongo



Where is the topology?


Independent, reusable!


Exchangeable.

Porting everything to Go.

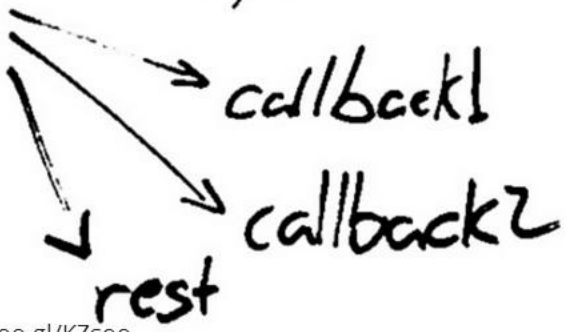
# Error checking...

```
v, err := f()
if err != nil {
    ...
}
```

Straightforward logic.

...and not this:

future1, future2 = zk.get_watch(p)



# How about one path:

```
data, ch := zk.GetWatch()
:
event := <-ch
:
```

Plus:

Static and Dynamic

"Static duck-typing"

Gophers →

Simple

Fast.

Good Library

gofmt

http://juju.ubuntu.com

Join us!

# Keith Rarick

Heroku

# What is Heroku?

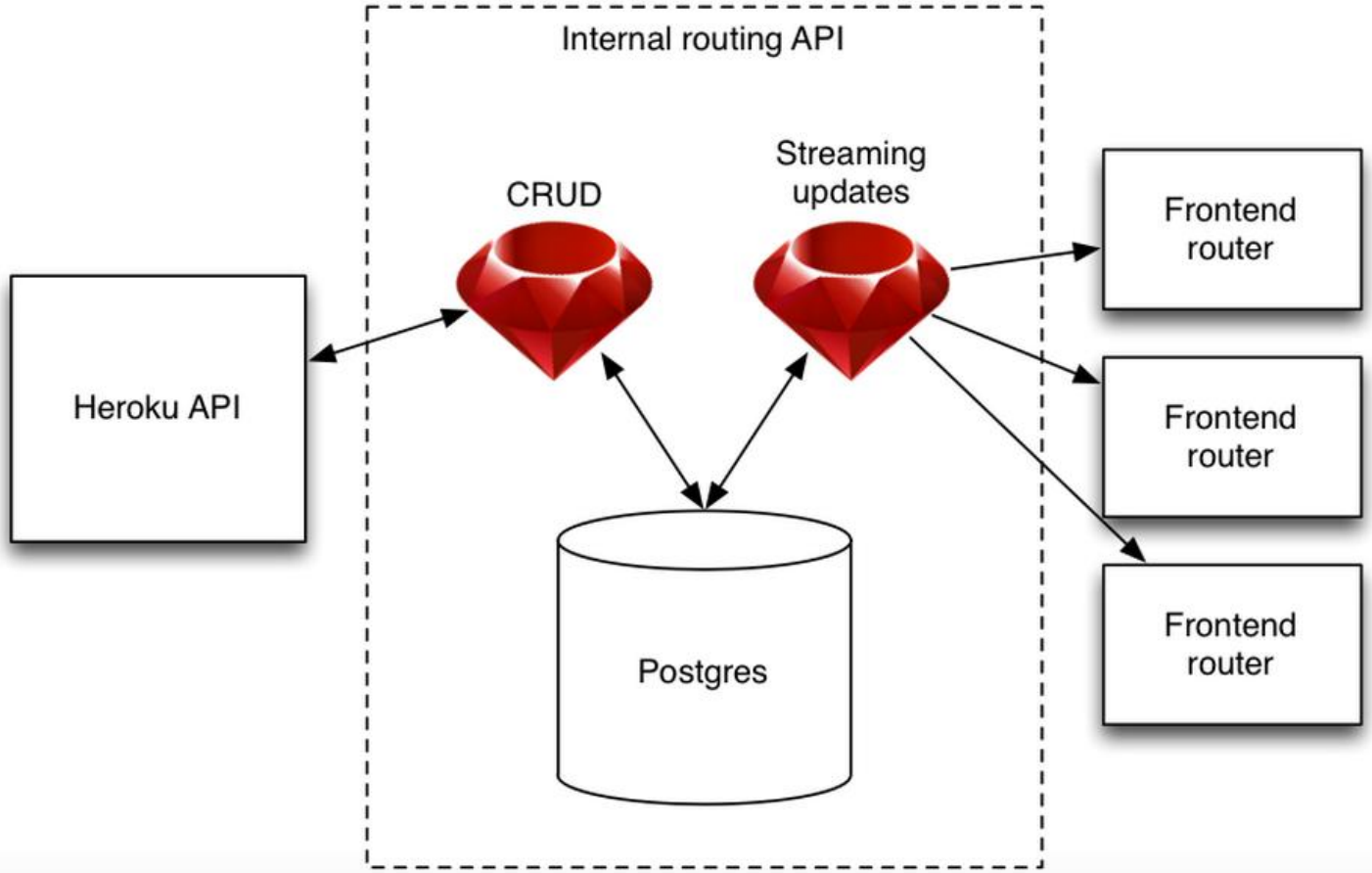

- PaaS for web applications in all languages
- 1.9 million apps
- About 85 employees
- Each active app runs one or more “dynos” – lightweight process containers

# Routing Table

- Changes whenever a Heroku dyno starts, stops, or moves
- For example, `heroku scale web=5`
- >1.9 million Heroku apps; a fraction of these are active at any time
- Plenty of churn

# Routing Table Service

- Originally all Ruby
- Accepts updates from the process manager
- HTTP router gets state from HTTP calls to this service
- Initial JSON dump to each HTTP router instance
- Followed by a stream of updates

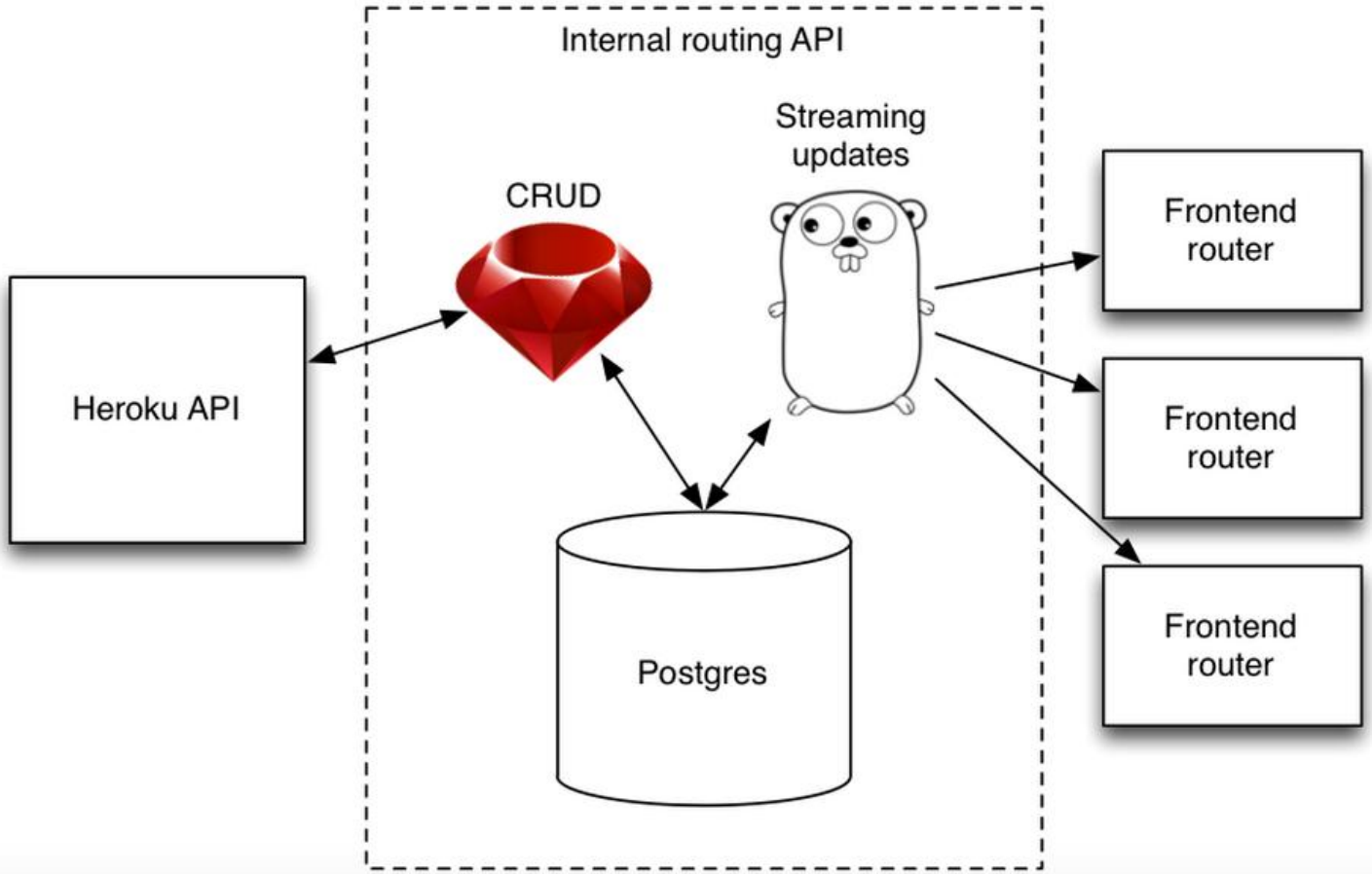

Internal routing API
CRUD
Streaming updates
Heroku API
Postgres
Frontend router
Frontend router
Frontend router

# The Problem

- Route lookups are fast – no IPC
- But bootstrapping a new router instance is slow
- Why? Generating the initial dump

# Enter Go

- Goal: drop-in replacement for the “streaming updates” piece
- Easy enough to play with during a single afternoon
- Working prototype after a couple of hours


Internal routing API
Streaming updates
CRUD
Heroku API
Postgres
Frontend router
Frontend router
Frontend router

# Slam Dunk

- 10× speedup
- Generating process (in Go) is no longer the bottleneck
- (Nor is the network)
- Easy to deploy and monitor, works great, so now in production

# Go at Heroku

- Started with experiments and side projects
- Now using Go for a handful of production services
- The list is growing

# Evan Shaw

Iron.io

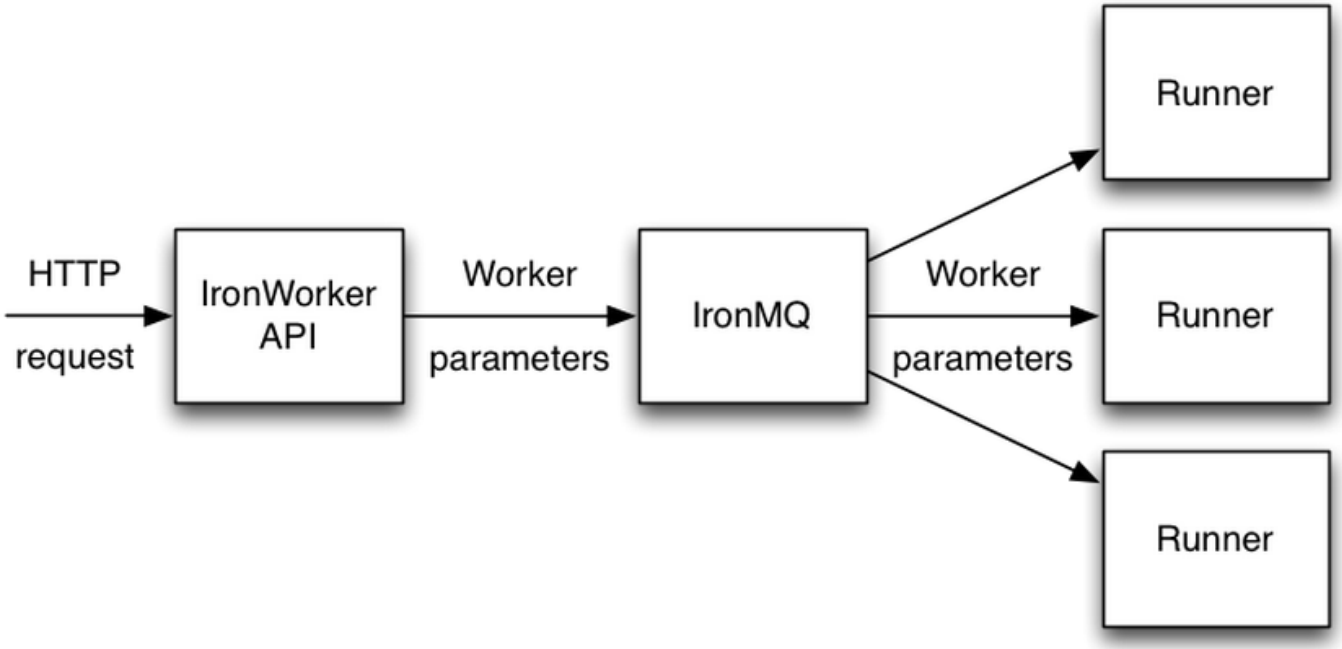

- Cloud application services
- Remove infrastructure worries
- Enable small teams to do big things


- Massively parallel worker platform
- API originally written with Ruby on Rails
- It was too slow
- Rewriting it in Go made it faster

# Other products

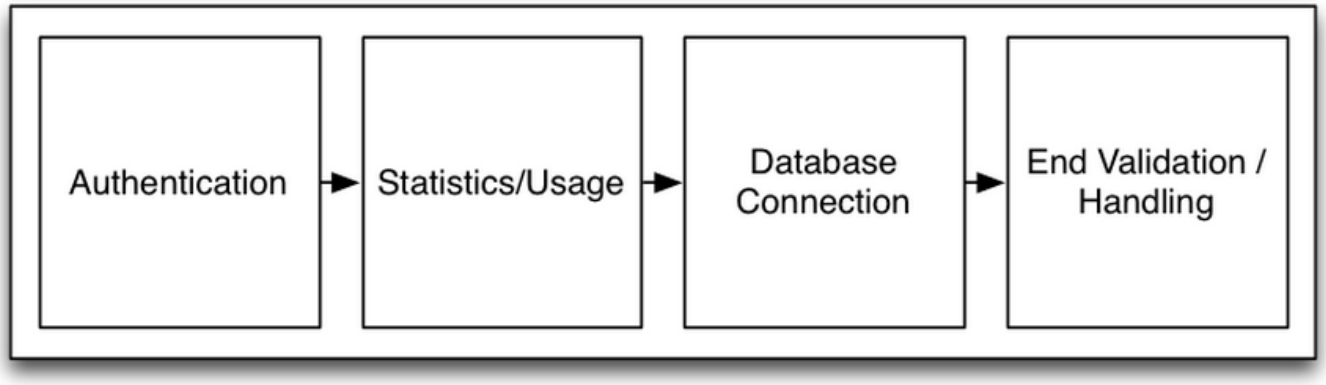
- IronMQ, scalable message queue
- IronCache, distributed key/value store
- REST/HTTP API and text protocol support

# The flow of IronWorker tasks



# Handling an HTTP request

- Compose HTTP handlers
  - Authentication
  - Database connection sharing
  - Statistics and usage



# Concurrency

- HTTP requests handled in separate goroutines by the HTTP package
- For text protocols, one goroutine per client connection
- Asynchronous I/O is automatic

# Concurrency as an optimization

- Non-critical operations can be moved to a separate goroutine
- Even less critical operations can be batched together in a dedicated goroutine

# Concurrency as an optimization

Before:

```go
func handler(w http.ResponseWriter, r *http.Request) {
  projectId := getProjectId(r)
  updateStatistics(projectId)
  // ...
}
```

After:

```go
func handler(w http.ResponseWriter, r *http.Request) {
  go func() {
    projectId := getProjectId(r)
    go updateStatistics(projectId)
  }()
  // ...
}
```

http://goo.gl/KZsoo

# Why Go?

- Fast
- Still reasonably expressive
- Rich standard library
- Not tied to a VM
- Awesome

# Patrick Crosby

StatHat


www.stathat.com @stat_hat

Track any stat with one line of code

# Online dating

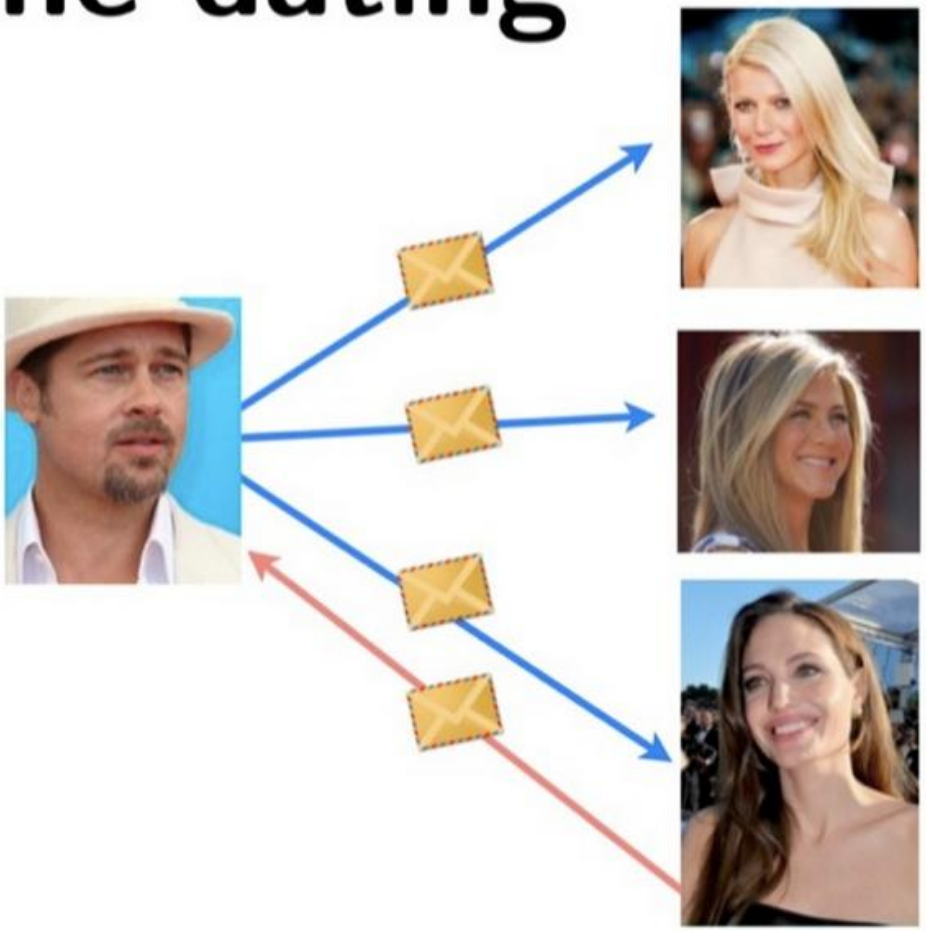
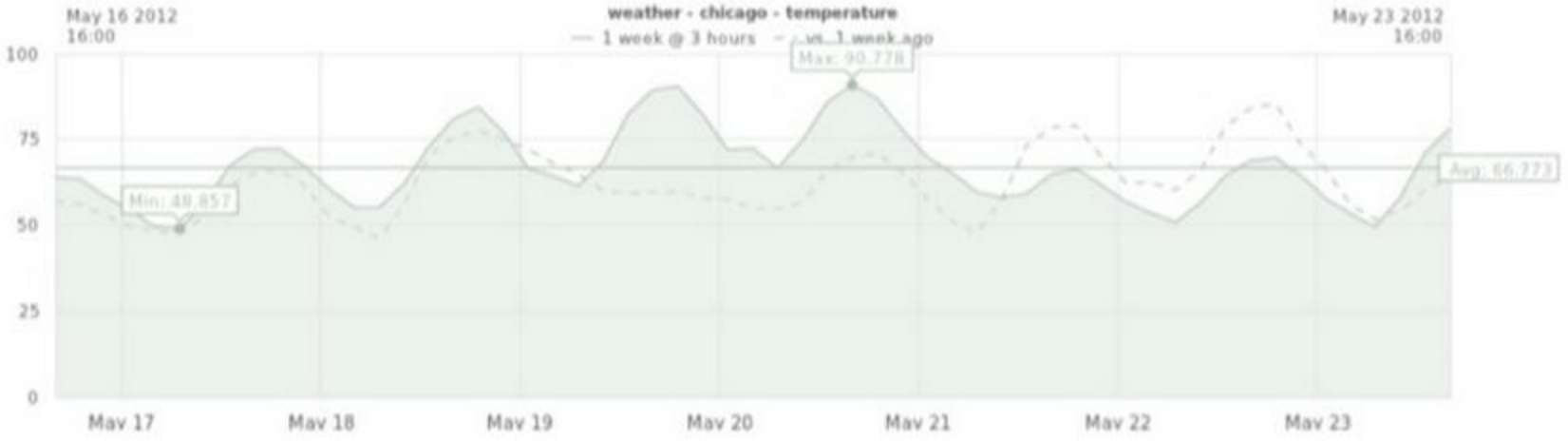


Image credits: Paltrow by Andrea Raffin, Pitt by Thomas Peter Schulzen, Jolie by Georges Biard, Aniston by Angela George

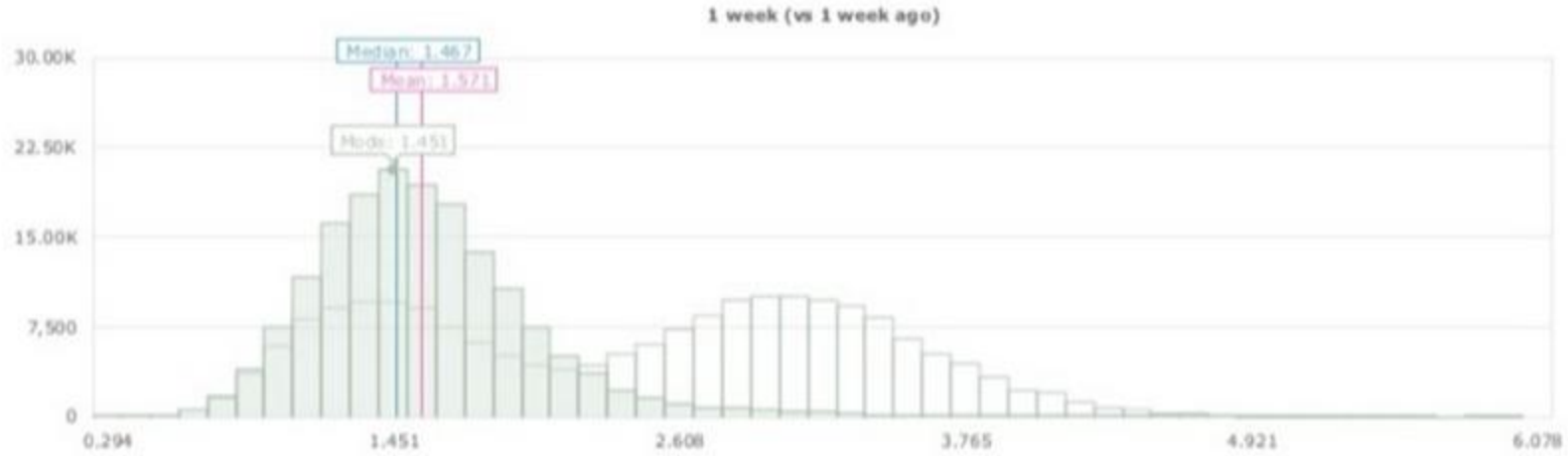
http://goo.gl/KZsoo

# 15 different languages

| | |
|---|---|
| Ruby | `StatHat::API.ez_post_count("reply count - female to male", "info@stathat.com", 1)` |
| Python | `stathat.ez_post_count('info@stathat.com', 'reply count - female to male', 1)` |
| Go | `stathat.PostEZCount("reply count - female to male", "info@stathat.com", 1)` |
| JavaScript | `_StatHat.push(['_trackCount', 'STATKEY', 1]);` |
| Node.js | `stathat.trackEZCount("info@stathat.com", "reply count - female to male", 1, function(status, json) {});` |
| HTML | `<img src="http://api.stathat.com/c?ukey=USERKEY&key=STATKEY&count=1" style="display:none;" width="1" height="1"/>` |
| iOS | `[StatHat postEZStat:@"reply count - female to male" withCount:1.0 forUser:@"info@stathat.com" delegate:self];` |
| PHP | `stathat_ez_count('info@stathat.com', 'reply count - female to male', 1);` |
| Java | `StatHat.ezPostCount("info@stathat.com", "reply count - female to male", 1.0);` |
| curl | `$ curl -d "stat=reply count - female to male&email=info@stathat.com&count=1" http://api.stathat.com/ez` |
| wget | `$ wget --post-data "stat=reply count - female to male&email=info@stathat.com&count=1" http://api.stathat.com/ez` |
| Lisp | `(stathat-ez-count "info@stathat.com" "reply count - female to male" 1)` |
| C# | `StatHat.Post.EzCounter("info@stathat.com", "reply count - female to male", 1);` |
| Perl | `stathat_ez_count("info@stathat.com", "reply count - female to male", 1);` |
| Lua | `stathat.ez_count("info@stathat.com", "reply count - female to male", 1)` |




# Time-series charts


1 week (vs 1 week ago)
Median: 1.467
Mean: 1.571
Mode: 1.451
30.00K
22.50K
15.00K
7,500
0
0.294
1.451
2.608
3.765
4.921
6.078

# Histograms


# Embed charts on any site

# Track anything

- Servers: load average, free memory, disk space, ping to google, number of processes
- Messages: size, time spent composing, reply counts, distance between users, age
- Requests: processing time, error occurred, number of calls, query time
- Other: number of jazz songs played, API calls, secs mobile app active, session length, weather, cups of tea, distance mouse travelled

ttp://goo.gl/K7sqo

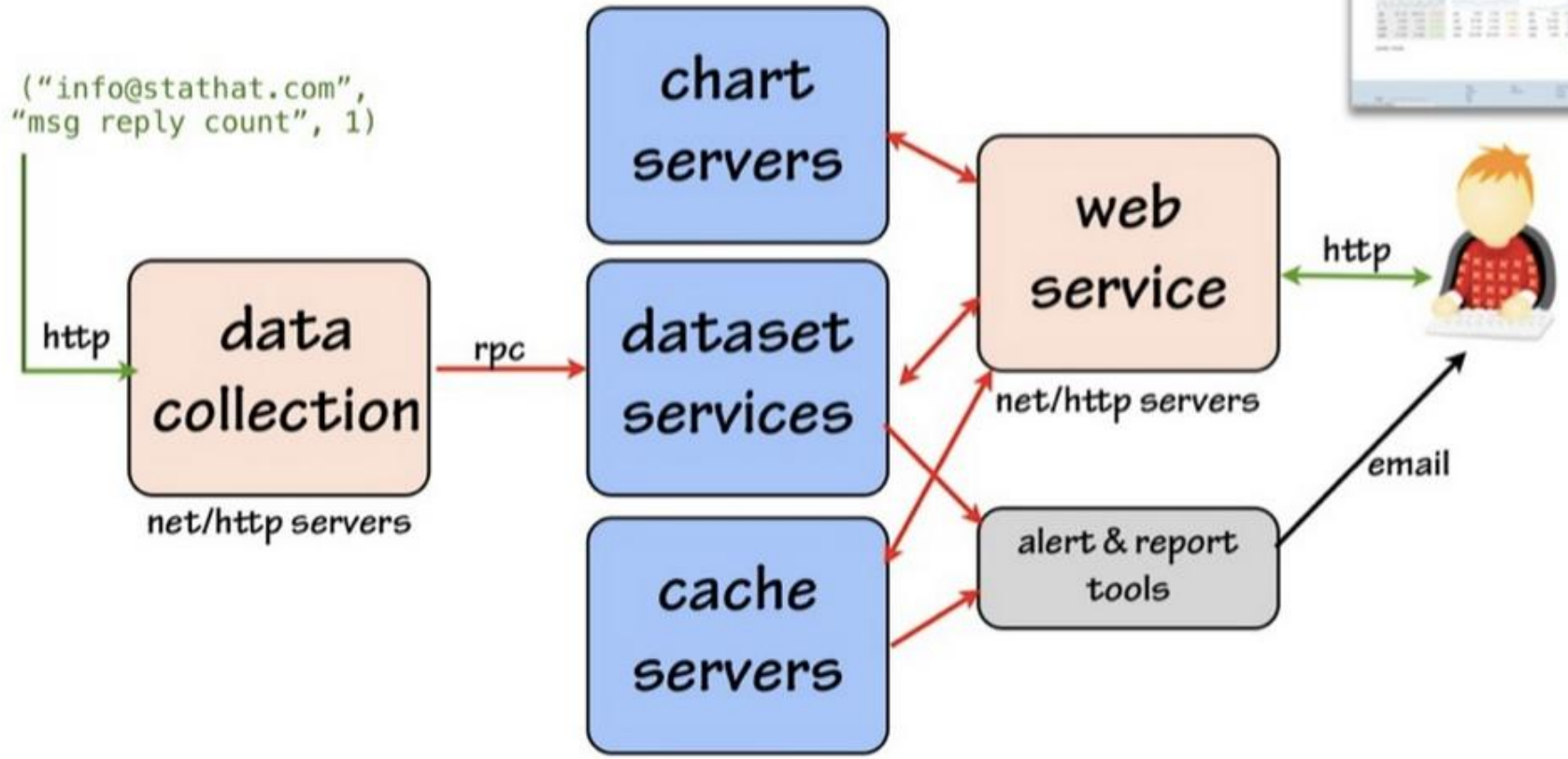
# 99.9% Pure Go

- web: net/http servers, html/template
- data collection: net/http servers
- backend: all services communicate using net/rpc

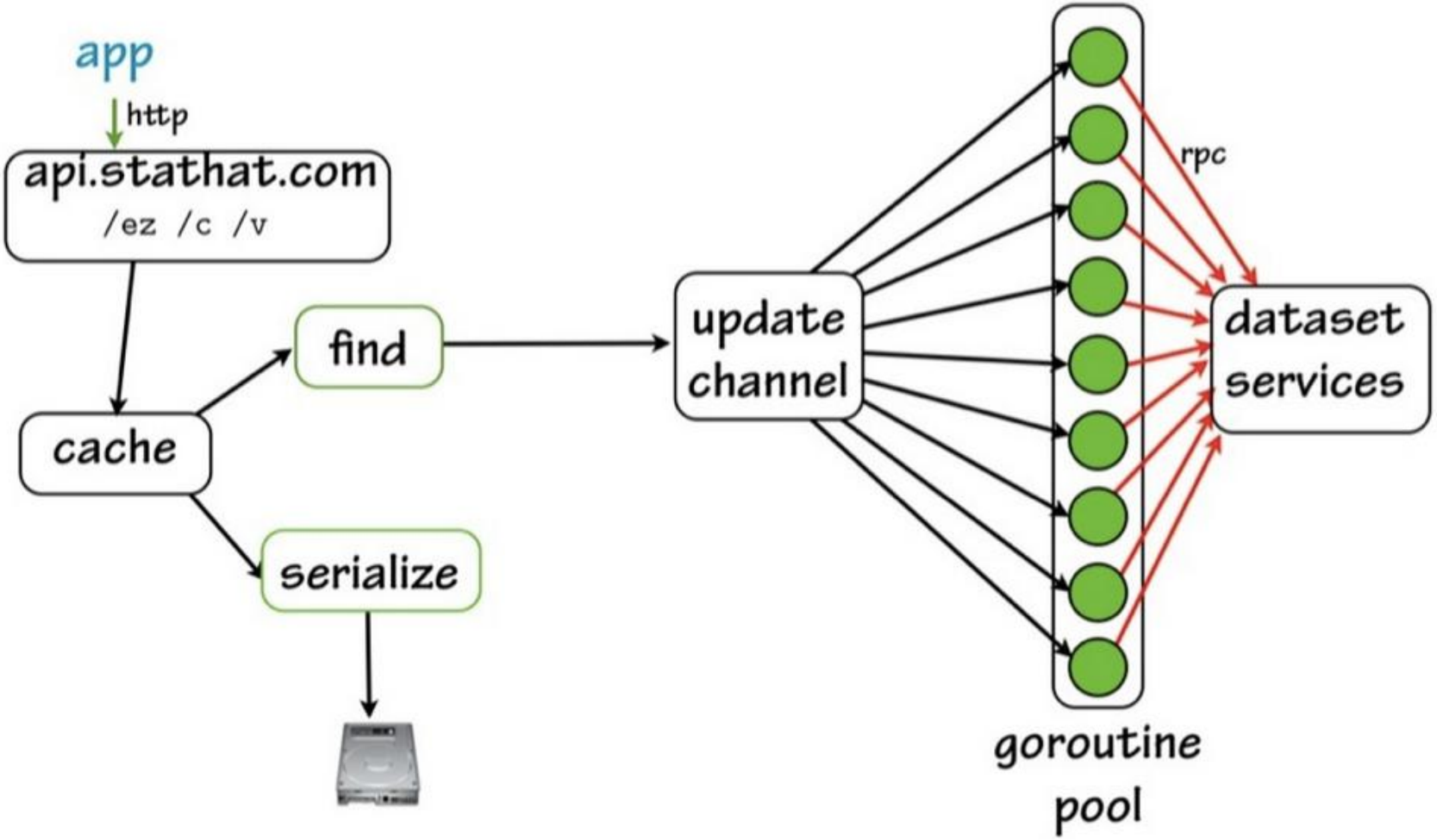
# Why Go?

- fast
- resource friendly
- easy to deploy
- fun

# Architecture overview



# Data collection service



# Deployment is easy

- Go binaries are self-contained
- No shared libraries, no gems, no dependencies
- go tool package install

# StatHat deployment

- git post-receive hook
- tests and compiles everything
- makes bundle
- rsyncs bundle to servers
- informs servers

# Thanks

Sign up:

**stathat.com/io2012**

Open source Go code:

**stathat.com/src**

Get in touch:

**patrick@stathat.com**

**@stat_hat**

# Q&A

Questions: http://goo.gl/KZsoo

Andrew Gerrand - Google
Gustavo Niemeyer - Canonical
Keith Rarick - Heroku
Evan Shaw - Iron.io
Patrick Crosby - StatHat